第 1 章

初识 Spring Boot

本章围绕 Spring Boot 项目的源码获取、编译和运行进行说明，简单介绍 Spring Boot 框架中一些模块的含义。

1.1 Spring Boot 源码编译

本节将对 Spring Boot 源码编译的过程进行说明。从零开始搭建 Spring Boot 源码阅读环境，搭建阅读环境需要 git、JDK8 及以上版本（Java 开发环境）、IDEA 编辑器和 Gradle 6.9 及以上版本。

1.1.1 Spring Boot 源码获取

关于 Spring Boot 源码获取需要在 GitHub 上找到 Spring Boot 项目仓库并将代码复制到本地，具体操作命令如下：

```
git clone git@github.com:spring-projects/spring-boot.git
```

当执行上述命令时有可能会出现如下异常信息（下文异常信息截取部分）：

```
error: unable to create file spring-boot-project/spring-boot-test-autoconfigure/
src/test/java/org/springframework/boot/test/autoconfigure/jdbc/
JdbcTestWithAutoConfigureTestDatabaseReplaceAutoConfiguredWithoutOverrideIntegrationTests.
java: Filename too long
```

当出现上述异常信息时表示文件名称太长了，这个现象在 Windows 系统中比较常见，原因是 git 调用的是 Windows 系统提供的旧 API 长度限制是 260，解决该问题只需要执行 git config --global core.longpaths true 代码。

当执行完成上述命令后就可以重新执行复制语句，将 GitHub 上的 Spring Boot 项目工程拉取到本地，拉取后在本地系统中会有如图 1-1 所示的内容。

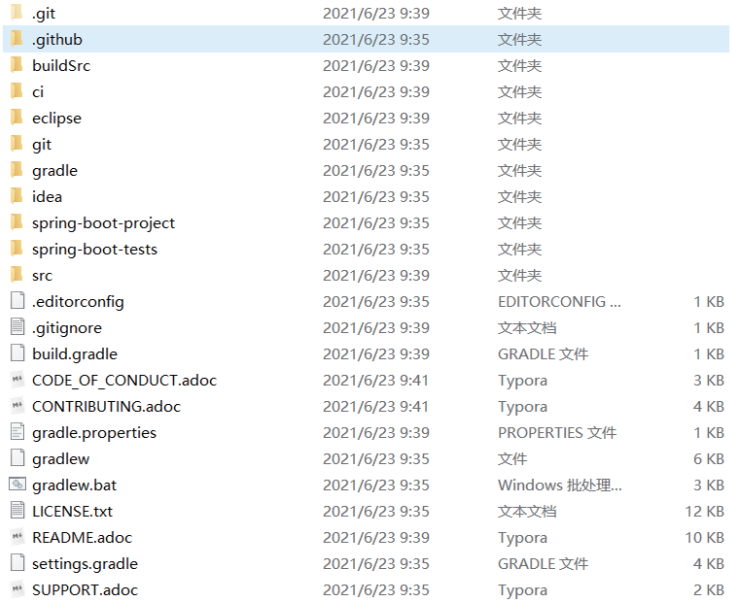

图 1-1　Spring Boot 本地仓库

接下来需要在该文件夹下打开 git bash 命令行工具，在命令行工具中输入如下命令：

```
git branch sh-2.4.6 v2.4.6
```

上述命令表示创建一个分支，该分支的源头是 v2.4.6。执行上述命令后需要执行 git branch 命令来确定是否创建成功。git branch 执行后命令行会输出如下内容：

```
$ git branch
* main
  sh-2.4.6
```

从输出内容中可以发现，sh-2.4.6 分支已经创建成功。接下来需要切换到该分支，具体切换命令如下：

```
git checkout sh-2.4.6
```

最后将这个分支推送到远程仓库，注意该远程仓库是个人远程仓库并非 Spring Boot 的官方仓库，具体推送命令如下：

```
$ git push origin sh-2.4.6
Total 0 (delta 0), reused 0 (delta 0), pack-reused 0
remote:
remote: Create a pull request for 'sh-2.4.6' on GitHub by visiting:
remote:      https://github.com/SourceHot/spring-boot/pull/new/sh-2.4.6
remote:
To github.com:SourceHot/spring-boot.git
 * [new branch]            sh-2.4.6 -> sh-2.4.6
```

1.1.2　Spring Boot 源码导入 IDEA

接下来将介绍如何将 Spring Boot 源码导入 IDEA 中，导入过程十分简单，只需要用 IDEA 将 Spring Boot 源码文件夹用 IDEA 打开即可。打开后界面显示内容如图 1-2 所示。

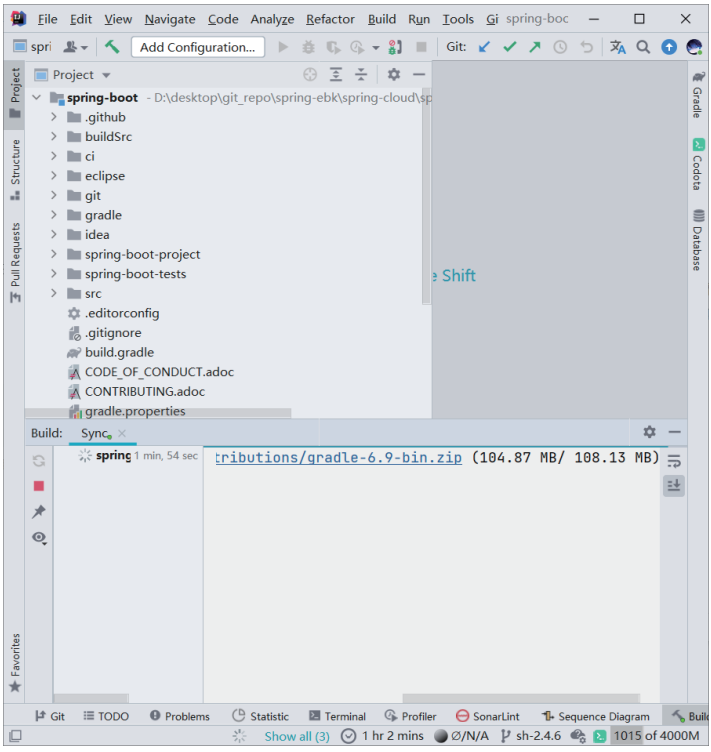

图 1-2　Spring Boot 导入 IDEA

在打开 Spring Boot 源码文件夹后 IDEA 会自动下载 Gradle 工具，当下载 Gradle 工具后还会继续下载 Spring Boot 项目所需要的依赖，此处预计花费一个小时左右的时间。当完成后控制台会输出如下内容：

```
    Download https://services.gradle.org/distributions/gradle-6.9-bin.zip finished,
took 1 m 55 s 190 ms (108.13 MB)
    Starting Gradle Daemon...
    Gradle Daemon started in 1 s 381 ms
    > Task :buildSrc:compileJava
    > Task :buildSrc:compileGroovy NO-SOURCE
    > Task :buildSrc:pluginDescriptors
    > Task :buildSrc:processResources
    > Task :buildSrc:classes
    > Task :buildSrc:jar
    > Task :buildSrc:generateSourceRoots
    > Task :buildSrc:assemble
    > Task :buildSrc:checkFormatMain
    > Task :buildSrc:checkFormatTest FROM-CACHE
    > Task :buildSrc:checkFormat
    > Task :buildSrc:checkstyleMain
    > Task :buildSrc:compileTestJava FROM-CACHE
    > Task :buildSrc:compileTestGroovy NO-SOURCE
    > Task :buildSrc:processTestResources
    > Task :buildSrc:testClasses
    > Task :buildSrc:checkstyleTest
    > Task :buildSrc:pluginUnderTestMetadata
    > Task :buildSrc:test SKIPPED
    > Task :buildSrc:validatePlugins FROM-CACHE
    > Task :buildSrc:check SKIPPED
```

```
> Task :buildSrc:build

Deprecated Gradle features were used in this build, making it incompatible with
Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.9/userguide/command_line_interface.html#sec:
command_line_warnings

BUILD SUCCESSFUL in 32m 7s

A build scan was not published as you have not authenticated with server 'ge.
spring.io'.
```

当看到上述信息后还需要等待十几分钟才可以完成 IDEA 的导入工作，导入后可以看到 IDEA 中的显示内容，如图 1-3 所示。

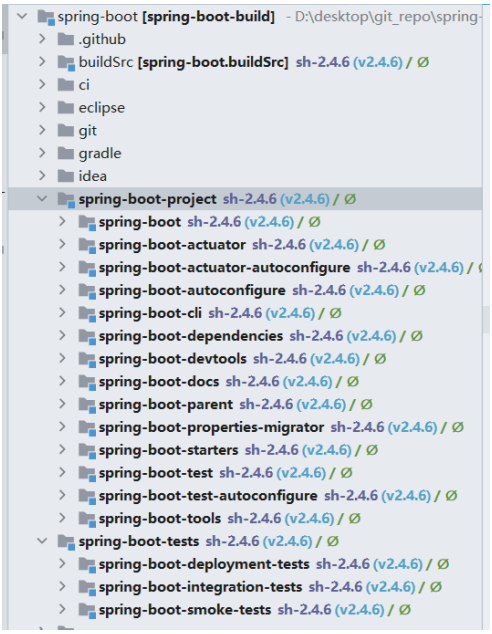

图 1-3　Spring Boot 工程明细

从图 1-3 中可以看到 Spring Boot 工程被分为了两大重要模块：

（1）spring-boot-project 模块主要用于存储 Spring Boot 框架的核心代码；

（2）spring-boot-tests 模块主要用于存储 Spring Boot 框架的测试代码。

在 spring-boot-project 模块中还根据不同的功能创建了多个模块工程，关于这些模块工程的说明如下：

（1）spring-boot：Spring Boot 项目的核心。

（2）spring-boot-actuator：Spring Boot 监控相关的内容。

（3）spring-boot-actuator-autoconfigure：Spring Boot 监控中自动装配相关的内容。

（4）spring-boot-autoconfigure：Spring Boot 自动装配相关的内容。

（5）spring-boot-cli：Spring Boot CLI 命令行工具。

（6）spring-boot-dependencies：Spring Boot 依赖工程。

（7）spring-boot-devtools：Spring Boot 开发工具工程。

（8）spring-boot-docs：Spring Boot 文档工程。

（9）spring-boot-parent：Spring Boot 项目的父工程。

（10）spring-boot-properties-migrator：Spring Boot 迁移工程。

（11）spring-boot-starters：Spring Boot 启动工程。

（12）spring-boot-test：Spring Boot 测试相关工程。

（13）spring-boot-test-autoconfigure：Spring Boot 测试相关的自动装配工程。

（14）spring-boot-tools：Spring Boot 工具工程。

在上述 14 个模块中，本书主要讨论分析的内容包括 spring-boot、spring-boot-actuator、spring-boot-starters 和 spring-boot-autoconfigure。更多关于 Spring Boot 子模块的说明可以查看 https：//docs.spring.io/spring-boot/docs/current/reference/html/using.html#using 进行了解。

1.2　Spring Boot 测试环境的搭建

前文已经通过 git 工具将 Spring Boot 的源码复制到本地操作系统中并且通过了编译，本节将讲述如何搭建 Spring Boot 的测试环境，即启动一个 Spring Boot 工程。在启动一个 Spring Boot 工程时可以采用 spring-boot-test 模块下 spring-boot-smoke-tests 模块中的内容。spring-boot-smoke-tests 模块主要用于进行冒烟测试，正好符合测试环境搭建的需求，在 spring-boot-smoke-tests 模块中含有多个工程，找到 spring-boot-smoke-test-web-static 工程，在该工程中找到 SampleWebStaticApplication 并运行它。运行后控制台会输出如下内容：

```
  .   ____          _            __ _ _
 /\\ / ___'_ __ _ _(_)_ __  __ _ \ \ \ \
( ( )\___ | '_ | '_| | '_ \/ _` | \ \ \ \
 \\/  ___)| |_)| | | | | || (_| |  ) ) ) )
  '  |____| .__|_| |_|_| |_\__, | / / / /
 =========|_|==============|___/=/_/_/_/
 :: Spring Boot ::

    2021-06-23 13: 38: 48.563  INFO 16852 --- [            main] s.w.s.SampleWebStat
icApplication          : Starting SampleWebStaticApplication using Java 1.8.0_252 on
LAPTOP-D7DF36F6 with PID 16852 (D: \desktop\git_repo\spring-ebk\spring-cloud\spring-
boot\spring-boot-tests\spring-boot-smoke-tests\spring-boot-smoke-test-web-static\
build\classes\java\main started by admin in D: \desktop\git_repo\spring-ebk\spring-
cloud\spring-boot)
    2021-06-23 13: 38: 48.566  INFO 16852 --- [           main] s.w.s.SampleWebStatic
Application         : No active profile set, falling back to default profiles: default
    2021-06-23 13: 38: 49.505  INFO 16852 --- [             main] o.s.b.w.embedded.
tomcat.TomcatWebServer  : Tomcat initialized with port(s): 8080 (http)
    2021-06-23 13: 38: 49.516  INFO 16852 --- [            main] o.apache.catalina.
core.StandardService   : Starting service [Tomcat]
    2021-06-23 13: 38: 49.516  INFO 16852 --- [           main] org.apache.catalina.
core.StandardEngine  : Starting Servlet engine: [Apache Tomcat/9.0.46]
    2021-06-23 13: 38: 49.601  INFO 16852 --- [            main] o.a.c.c.C.[Tomcat].
[localhost].[/]       : Initializing Spring embedded WebApplicationContext
    2021-06-23 13: 38: 49.602  INFO 16852 --- [           main] w.s.c.ServletWebServe
rApplicationContext : Root WebApplicationContext: initialization completed in 981 ms
    2021-06-23 13: 38: 49.865  INFO 16852 --- [           main] o.s.b.a.w.s.WelcomePa
geHandlerMapping    : Adding welcome page: class path resource [static/index.html]
```

```
    2021-06-23 13: 38: 49.973  INFO 16852 --- [           main] o.s.b.w.embedded.
tomcat.TomcatWebServer  : Tomcat started on port(s): 8080 (http) with context path ''
    2021-06-23 13: 38: 49.983  INFO 16852 --- [           main] s.w.s.SampleWebSta
ticApplication          : Started SampleWebStaticApplication in 2.051 seconds (JVM
running for 3.34)
    2021-06-23 13: 39: 05.236  INFO 16852 --- [nio-8080-exec-1] o.a.c.c.C.[Tomcat].
[localhost].[/]       : Initializing Spring DispatcherServlet 'dispatcherServlet'
    2021-06-23 13: 39: 05.237  INFO 16852 --- [nio-8080-exec-1] o.s.web.servlet.
DispatcherServlet       : Initializing Servlet 'dispatcherServlet'
    2021-06-23 13: 39: 05.237  INFO 16852 --- [nio-8080-exec-1] o.s.web.servlet.
DispatcherServlet       : Completed initialization in 0 ms
```

输出上述内容表示启动成功。接下来需要通过浏览器访问 http：//localhost：8080/，访问后浏览器展示内容如图 1-4 所示。

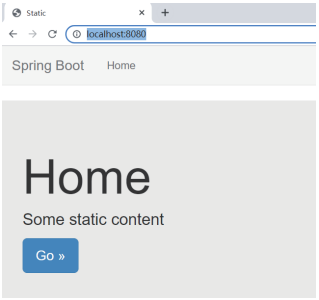

图 1-4　Spring Boot 测试首页

在 spring-boot-smoke-tests 模块下还有不同技术的测试用例可以作为相应技术的测试环境，本节就不对每个技术模块进行说明了，有兴趣的读者可以自行查看各个技术模块。

1.3　Spring Boot 编译后可能遇到的问题

本节将对 Spring Boot 编译后可能遇到的问题进行说明，并提供相应的解决方法。

在启动 smoketest.propertyvalidation.SamplePropertyValidationApplication 的时候可能会出现启动失败的问题（并非特指 SamplePropertyValidationApplication，其他应用启动类也有可能出现此问题），具体现象是普通启动在 IDEA 编辑器中并未出现问题，当通过 debug 启动时在 IDEA 中出现了报错。关于报错的异常堆栈信息如下：

```
    java.lang.NoClassDefFoundError: kotlin/Result
        at kotlinx.coroutines.debug.AgentPremain.<clinit>(AgentPremain.kt: 24)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at  sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:
62)
        at  sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.
java: 43)
        at java.lang.reflect.Method.invoke(Method.java: 498)
        at  sun.instrument.InstrumentationImpl.loadClassAndStartAgent(InstrumentationImpl.
java: 386)
        at  sun.instrument.InstrumentationImpl.loadClassAndCallPremain(InstrumentationImpl.
java: 401)
    Exception in thread "main" FATAL ERROR in native method: processing of -javaagent
failed
```

关于这个问题笔者在 Spring Boot 的 GitHub 仓库提了一个 Issues，具体地址是 https://github.com/spring-projects/spring-boot/issues/27531。在和 Spring Boot 项目成员交流后得出了如下解决方案。在 IDEA 中打开设置界面，依次选择 Build，Execution，Deployment → Debugger → Data Views → Kotlin 选项，进入之后会看到如图 1-5 所示的内容。

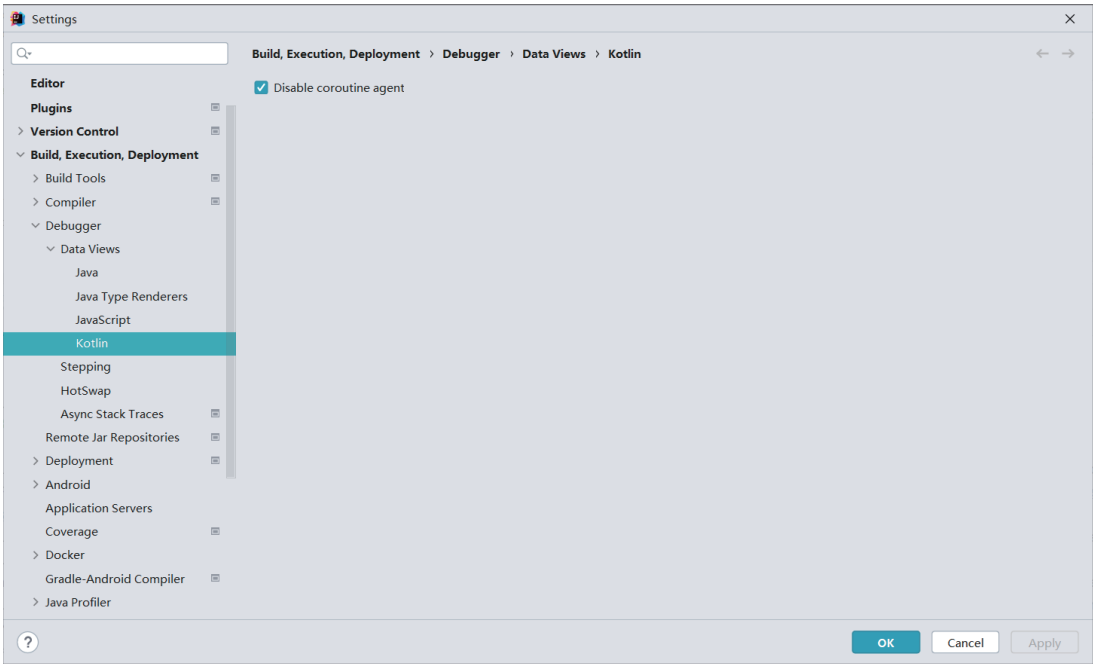

图 1-5　IDEA Kotlin 配置

注意在 IDEA 默认状态下 Disable coroutine agent 是没有选中的，此时需要选中此选项，修改成功后单击 Apply 按钮和 OK 按钮完成配置。通过上述配置处理后再进行 debug 启动就不会遇到本节最开始的问题了。

本章小结

本章中对于 Spring Boot 框架的源码获取和编译做出了详细的说明，在整个搭建过程中最为麻烦的问题有两个，第一个是在 Windows 系统下文件名过长的问题，第二个是获取 Spring Boot 依赖的时间问题。这两个问题解决后 Spring Boot 源码阅读环境的搭建就很容易了。本章基于搭建的源码环境还启动了一个 Spring Boot 应用来证明 Spring Boot 环境的可用性。